FSA Modernization Partner

United States Department of Education

Federal Student Aid

Integrated Technical Architecture
Performance Testing Best Practices Guide

Task Order #69

Deliverable # 69.1.4

Version 2.0

June 27, 2002

Table of Contents

1 Executive Summary

The Integrated Technical Architecture (ITA) Performance Testing Best Practices Guide is a document that will guide FSA application teams through the process of performance testing their FSA web based applications. Using industry standard tools and proven methodologies, application teams can prove scalability and availability goals specified by FSA application requirements, as well as detect load based problems that go undetected in functional and integrated testing.

The goal of an application performance test is to focus on three core areas:

1) Application scalability
2) Application performance
3) Infrastructure assessment

With these goals in mind, an application team can determine the degree of tuning required, decide on capacity limits, forecast availability and maintenance plans, and determine the maximum system performance with the optimum configuration. By determining these factors, production migrations become less complicated and support costs are dramatically reduced.

1.1 Purpose

The purpose of this document is to collect and document the Best Practices that have been identified and used for performance testing within the WebSphere environment at FSA using Mercury Interactive's LoadRunner. These practices are assembled from recommendations provided by Mercury Interactive, IBM and Accenture through their in-depth experience with E-Commerce and its implementation within Java based Application Servers.

1.2 Intended audience

The ITA Performance Testing Best Practices Guide is for technical architects, designers, application developers, testers and performance administrators who are responsible for performance testing Java-Based solutions within the SFA WebSphere environment. The document assumes that the reader is knowledgeable in Java and the WebSphere Application Server and has access to the Java 2 Software Development Kit (JDK) and WebSphere reference documents.

1.3 Background

In assisting with the performance testing of multiple FSA applications, the ITA team determined the necessity of collecting the Best Practices white papers on performance testing and administration of applications that run within the FSA WebSphere environment. This allows FSA to benefit from previous work done at other WebSphere implementations that have carried out benchmarking and performance testing work to identify best practices. These best practices help FSA design their applications to extract maximum performance from WebSphere and to ensure quality and scalable web sites.

1.4 Scope

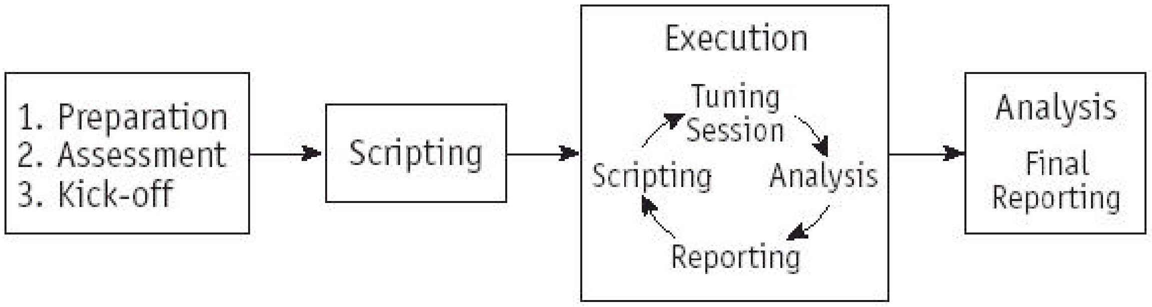
These guidelines will provide specific examples and recommendations for running performance tests in a highly analytical and efficient process. This document does not include the application code level best practices. ITA best practice Release 2 has application code level best practices (eg. session, servlet, Java code best practices).

1.5 Assumptions

The current WebSphere environment at FSA includes WebSphere 3.5.5, its associated Web Server (IBM HTTP Server 1.3.12), Oracle databases (8.1.7), Java Development Kit (1.2.2), LoadRunner 7.01, and JProbe 3.0. These recommendations may change with later releases of these products.

2 Overview

There are four phases in doing the application performance test using Load Runner: Planning, Scripting, Execution, and Analysis and Tuning. The following diagram shows the performance test life cycle.



The first stage, which is the planning stage, includes test preparation, understanding the application architecture, and a kick-off meeting. The second stage is the scripting stage where the load runner scripts are created. The third stage is the execution stage where the performance test is executed. The execution stage has multiple iterations where the test is run, the system is tuned, the test is analyzed, and the scripts fixed if necessary. The fourth stage is the analysis stage where the test results will be analyzed and the reports created. Each of these steps and its related best practices is discussed in the following sections of this document.

The ITA team also utilizes JProbe to do application profiling. ITA recommends that the application team should profile their application before the load test. Application profiling best practices can be found in Section 7 of this document.

3 Planning

In the planning stage includes understanding the application architecture and establishing the performance test objectives. Thus, the performance test planning should run parallel to the system development. The performance test plan document should be produced at the end of the planning stage.

3.1 Performance test plan

ITA and the application team should work together on this document. This document should include a diagram and a written summary that details the performance test environment and any external interfaces that might exist. External interfaces could be an Oracle database, MQSeries on the mainframe, a search engine or an email engine. This diagram is used to ensure that the performance test does include all details that production is likely to feature. It should also outline the test goals, test dates, test scenarios, and business processes.

3.2 Identification of the different run-time scenarios

Users do different things on a web Site. The goal of this function is to identify a reasonable number of processes that users will execute on a web site so that the real world is simulated. Collectively, scenarios for the application will address most functions expected by a web site that is undergoing performance testing. To gather the different scenarios, the following critical questions need to be asked.

- What business functions are critical to the function of the application?
- What actions do end users most frequently use?
- Who are the users?
- Which business functions are database intensive?

Additional business process selection is driven by the impact of the processes on different system components. In most systems, 80%of the load is created by 20% of the transactions. In a typical system, repetitive business processes create the most significant load. Normally it is optimal to select three to ten business transactions that reflect critical end-user actions. Critical business functions for a Web site may include:

- Site browsing
- Searching
- Registration

Another factor to consider and document at this point is business process length. Each scenario needs a time associated with how long a user takes to complete a business process. These times are usually kept short, under five minutes, in comparison to actual user session length on a web site. This is because a real user usually performs a combination of the business processes on the actual site, as it is assumed that many short scripts utilized together represent end users that may be on the system longer. Using the previous example:

- Site browsing (5 minutes)
- Searching (3 minutes)
- Registration (3 minutes)

3.3 Percentage of concurrent users traversing each business process

To accurately simulate real world conditions on a web server, the performance plan should have an understanding of how many users do which business process. An example would be:

- Site browsing (50%)
- Searching (15%)
- Registration (35%)

If previous versions of web sites exist then this information can be gathered via a Web analysis tool such as Web Trends.

3.4 Trends

It is important to recognize any peak periods that might require extra capacity or off-peak periods where capacity can be withdrawn. An example would be if an application had a steady state that lasted eleven months but a peak period that lasted one month but was three times the steady state. It is important to test at the peak for capacity limits and test at steady state for capacity planning.

3.5 Stress Test vs. Load Test

Load testing is an evaluation of system performance under normal conditions, which includes testing the targeted number of concurrent users supported by the application’s configuration to meet the application goal. Stress testing is the determination of the application’s limitations and breaking point. It is imperative that both stress testing and load testing are performed on an application.

An example of load testing is if an application that may have a requirement for supporting “250 concurrent users.” This requirement can be verified by a load test which ascertains whether or not 250 concurrent users can be supported. If the application meets the requirement then the goal is met.

Stress testing, on the other hand, explores the limitations of an application. Using the requirement from the previous example – 250 concurrent users – stress testing would determine the number of users in excess of 250 that the application would support. Applications could reach their breaking point at 300 users or 1000 users. It is therefore significant to establish and understand the limitations of each application, as stress testing is closely linked to the notion of scalability. An application’s scope for scalability can be deduced from the results obtained from application stress testing.

3.6 Measurable Goals

The performance test plan should include goals that state capacity and response time limits that need to be achieved during the performance test. These goals can be operationally oriented (e.g., the ability to maintain 3000 concurrent users) or business-oriented (e.g., ability to handle 1000 invoices an hour). Either way, goals should be written in such a way that they are measurable.

An example of a test goal would be:

The performance test for the Portals application will verify that 200 concurrent users are able to perform a registration with response times less than 30 seconds.

There should be at least one goal for each business process identified.

3.7 Response Time

Response time is defined as the duration of a server to response to a user request. If an application has a slow response time then it risks losing its customers. Web sites that permit users to transact business must offer their services in a way that meets the users' needs. This means "performance" is usually measured in response time to a user's request. In a web application the major factors that contribute to download time are page size in kilobytes, number and complexity of items, number of servers accessed, and whether SSL is used. Response time can be ranked as following:

Seconds to load	Ranking
Less than 10	Excellent
10-15	Very good
15-20	Good
20-25	Adequate
25-30	Slow
Over 30	Unacceptable

This is just a general guideline. An application might have a requirement where it is expected to load a search page in less than 5 seconds or some reports might take 3 to 4 minutes to load which might be acceptable.

An application should consider their user types (LAN vs. modem connection). If most of the users connect to the site from a typical dial-in connection then design must optimize for them to assure their visit is successful. Load Runner has a capability to simulate the users coming via the dial-up connections.

3.8 Business Process

For an effective performance test, the test team needs to choose meaningful transactions to measure. The business processes should be broken up into logical steps, with each step being equivalent to a transaction. The transactions that are defined should map closely to the objectives of the tuning session. For example, if one of the objectives is to ascertain the average response time required in searching for an item from a database, then it is necessary to have a transaction that measures this step in the business process.

Business analysts or functional experts should be consulted to determine where the performance measurements are needed. Some common transaction measurements are: user actions that insert, update, search the database, heavy graphic manipulation, actions which cause user authentication, actions which create and hold server connections, audio or video streaming, file downloads, or any other processes that may be considered system resource intensive. These transactions are usually tied to Servlet calls (depends upon the application architecture). Tools like Load Runner usually present transaction response times so that identified transactions may be tracked.

3.9 Schedule

It is essential to review the following setup and scheduling guidelines before the tuning session begins: If the application is planning to use the ITA team to do their performance test then the application team needs to get on the ITA team performance test schedule. The ITA team performance test schedule gets booked very quickly so the application team lead needs to set up a meeting with the ITA team lead to get on the ITA schedule so that ITA can determine all the performance test hardware, software, and environment for a particular application. During the preparation phase it is critical to consider any data preparation necessary such as emptying database tables, and restarting web servers.

4 Scripting

ITA uses the functional test script provided by the application team to generate the load runner scripts. Functional scripts outline step-by-step actions that a user needs to take to complete a business process. Functional experts who understand the business processes that are being scripted should be available to answer questions and resolve any process issues. The script creation process is one of the most important steps in the entire performance test process. ITA uses Load Runner Virtual User Generator (VuGen) development kit to record the performance test scripts. Once a script is recorded, it is necessary to make certain modifications to support dynamic values, volume data, data correlations, parameterization, transaction demarcation iteration looping and other virtual users options to make them behave like real users.

4.1 System Stability

Before creating the scripts the application must be functionally stable. Application changes should be kept to a minimum during the scripting process. Changes in the user interface, server functionality or database data during the scripting process may cause working scripts to fail.

For example, if requirements change and an information screen is altered so that a social security number is a required field, then a script which has been previously recorded without the social security number will no longer work when replayed against the new version of the application. This can also be true with changes to business rules or the underlying database. A script may also rely on preloaded data that is expected to be in the system. For example, if a script orders a product, such as a laser printer, and the data in the product table is deleted, then the script will fail when it attempts to order a part that does not exist. While the application is undergoing performance testing, frequent system changes can cause severe impact to the project timeline and should be avoided.

4.2 User Pacing

User pacing is defined as an average time taken for users to finish a business process. User pacing is important in order to simulate real-world conditions. If the system is expected to support 1,000 concurrent users doing work, the test team must ask, "at what pace do they perform that work?" An average user may stop and read product descriptions, the home page and other similar pages for 30 to 45 seconds before the next mouse click. It may take a user an average of 60 seconds to enter the information on a registration page before pressing the submit button. The test team can vary this pacing (known as think times) in the script from static think times, such as 5, 10, or 15 seconds, or set them as random, with a range such as "wait 5 to 15 seconds from when the home page appears to the time the search link is clicked."

5 Performance Test Execution

Once the performance test scripts are ready then the next step would be to execute the test. Performance testing is typically scheduled into three-hour tuning sessions. These are normally arranged during off-hours or low-traffic hours to minimize impact on users during normal business hours, if the performance test and production environment are using the same resources. Otherwise, a performance test can be performed during business hours as long as it does not share the production environment and resources.

Usually a conference call is set up so that all teams may communicate during the test. Since performance testing is a team effort all parties are expected to have representatives on hand during the performance test (e.g., Database Administrator, Network Administrator, System Administrator, Developer, Business Analyst or Functional Expert, Web and Application Server Administrator, and Technical Architect etc.). A small application must have at least three performance tests, but it is not unusual to have more tests to resolve problems, especially for larger applications. Several test practices apply to implementation, which are discussed below.

5.1 Tuning Session

Tuning is the process of performing enhancements to an application or configuration to improve response time. A tuning session allows sufficient time for proper user ramp-up, stabilization, real-time analysis and bottleneck identification. This is the point where the proper planning, script creation, and data preparation is realized. If fixes are applied during a test, they should be applied incrementally (i.e., one at a time) and then verified. Multiple fixes could cause multiple problems if applied simultaneously.

5.2 Plan on a Infrastructure Test to verify network and server capacity

The first area examined during the testing process is the infrastructure. The infrastructure precedes the application architecturally so any performance issues in the infrastructure will impact the application. This approach is essential since it allows the test team to certify the infrastructure before concentrating on the system being tested. The testing team essentially divides the system, allowing proper diagnosis of any performance bottlenecks for all components from the end user up to, but not including, the web servers. Infrastructure testing will establish the capacity and stability of the load balancer, firewall, routers, switches, network cards, ISP service level and bandwidth capacity. The infrastructure test is usually accomplished by generating a maximum number of users to retrieve a single GIF or JPEG file. If problems are found, it may take multiple iterations of infrastructure tuning to diagnose, solve and verify that the proper fix has been applied.

5.3 Establishing Baseline

Once the infrastructure test has been conducted successfully and the bandwidth capacity is verified, server and application testing can begin. Application performance testing consists of multiple tests in order to tune the components inside the firewall. The first test is a baseline test to establish the current performance capacity and scalability of the system. The typical scenario (business process mix) is used in this stage and the number of running users is slowly increased until performance becomes unacceptable. The back-end systems, such as Web servers, application servers, database, legacy, etc., are closely monitored at this point. Memory, paging/swapping, CPU, processes, connections, queues, cache, database buffer pools, and other system usage must be watched to identify trends, subtle inefficiencies, and abnormalities that can affect end-user response time as the number of users are increased.

5.4 Ramping up users

Once the infrastructure test is completed and the baseline is established, normal application testing and tuning begins. The ramp-up in the number of users that is running initially must be done in a consistent and repeatable manner. The systematic increase of users cannot be rushed or the initial baseline test results may not be valid. The number of users must be gradually increased so as not to cause unrealistic system spikes in performance. A typical ramp up strategy is as follows:

Run one user of each script and carefully record the response times for each transaction. Allow each user to make more than a single iteration. This provides a few data points for each transaction and allows for proper transaction averaging. This will establish a baseline for performance under optimal conditions.

Ramp-up users 10% in one minute intervals or LoadRunner has the facility to ramp-up users automatically at the rate of 50 users every 90 seconds. Testers are expected to maintain vigilance in observing performance statistics while the test is underway.

Repeat this process slowly, increasing the number of users until the team finds a proper “step level” for the user ramp up. This is different for each system, but experience indicates that each step be no more than 5 to 10% of the desired capacity. For example, a system that is expected to handle 1,000 concurrent users might be increased in 50-user increments. If the baseline established indicates that the system can handle a higher user increment, then this increment should be used on subsequent tests.

LoadRunner has an option whereby user ramp-up can be stopped, if a system is not acting as expected or if performance degradation is observed. Under these circumstances the ramp-up should be stopped and the problems that led to decreased performance should be analyzed. This does not imply that the performance test itself should be stopped, only that performance degradation should be addressed.

Unless the particular test scenario is deliberately attempting to test a sudden increase in users, such as the opening bell on the stock market or a similar event, users should always be increased gradually. Once the baseline is established and the system has gone through a series of tuning cycles, these kinds of stress tests can be benchmarked.

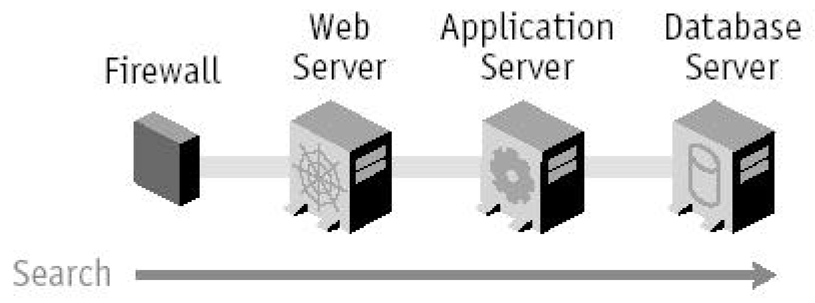
6 Analysis

After executing the performance test it will help to identify the performance bottlenecks. After each test cycle some analysis should be done and the problems fixed before running the next cycle. ITA team follows the following performance test best practices.

6.1 Server Level Bottleneck Identification

One reason for doing a performance test is to identify bottlenecks that only exist in a load driven scenario. Each transaction is measured during a tuning test: home page loading, links, data submits, static pages, dynamic pages, data searches, report creation, logins, etc. If the test team sees particular transactions beginning to have performance degradation while other transactions continue to perform well, they can refer the transaction to “component-level mapping” that they performed during the planning stage to assist in isolating the bottleneck. The performance test team first must determine which server or interaction between servers may be causing the bottleneck before they can isolate the particular tuning issue. They do this using specific tier-level stressing to determine if the tier is performing at maximum capacity. To gather this information, the tuning specialist can utilize server statistics, application logs, real-time performance monitors and other data that will lead them to the server tunable that needs adjustment. If a particular tier, such as Web server cluster, is not performing well, they can drill down to do individual component tuning and optimization. In the FSA ITA environment, WebSphere has a Resource Analyzer, which presents the Java Development Kit’s 1.2.2 performance metrics on Memory, Orb connections, servlet and data-source statistics.

For example suppose the performance test team notices that the “search” function is taking quite a bit of time, and that all the hardware metrics for the entire system are under 50% utilization. Since the search function utilizes the entire system the performance test team would narrow the test.



The performance test team can add a specific Web server test now to determine if the Web server is slow to respond. If the web server response is faster than the “search” response then the team would deduce that the Web server and the firewall are not the culprits. The team would replace the Web server test with an application test and run the performance test again. If performance from the application server is better than the “search” then the team would deduce that the firewall, web server, and application server can be excluded. Now, the performance test team would create a different database test to determine if it is in fact the database itself or just the “search” function. Although many architectures are much more complicated and require many more steps, this is the fundamental approach to tier-by-tier bottleneck isolation.

6.2 Use the Resource Analyzer

WebSphere has a tool named Resource Analyzer, which presents the Java Development Kit's 1.2.2 performance metrics on Memory, Orb connections, servlet and data-source statistics. These statistics are helpful in understanding if a memory shortage is occurring or a database bottleneck has occurred. The resource analyzer data can be logged in real time or replayed into a log for later play back. The resource analyzer captures Servlet and data source level information.

6.3 Use "verbosegc" in the Application Server command line

The Java heap size is the amount of memory dedicated to the Java Virtual Machine (JVM) for application run time execution. It is critical to the performance of the application whether the heap size is properly sized for the application that is being executed. If the number of users is too great for the JVM, then the application will consume too much memory and the WebSphere application server will not be able to continue.

To determine the heap size, a baseline stress test needs to be run, with *verbosegc* indicated in the JVM command line. This parameter in the command line informs the JDK to write out the amount of memory freed during a garbage collection session. A garbage collection session is the JDK sweeping through all of the objects that been created within the heap and determining whether any objects can be freed so that the memory is returned to the heap for reuse. If after a full garbage collection, the collector can return 60% to 80% of the heap as free memory then the amount of memory allocated to the heap is correct. If the amount of memory is greater than 80%, administrators should consider lowering the heap size. If the amount of memory freed is less than 60%, administrators should raise the heap size. If the amount of free memory under maximum load is greater than 90% then administrators should consider lowering the heap size.

6.4 Set the maximum Heap size equal to the minimum Heap size

The heap size can be specified in terms of a minimum size and maximum size. The minimum and maximum size configuration allows the Java heap to increase in size up to the maximum specified. Since performance testing should be focused on application runtime performance and tuning, the minimum heap size is set to equal the maximum heap size.

6.5 Monitor the Web Server Processes

IBM Http Server (IHS), which is a derivative of the Apache web server, is the standard FSA Web Server for the ITA project. IHS is a process driven Web Server, which means that as the load increases on the Web Server, Apache will use a proprietary algorithm to increase the number of listening processes to handle the workload. Apache has a configuration ceiling for this function so that the web server doesn't overuse the hardware that supports it. The configuration parameter is called MaxClients and is configured within the file *apacheroot*/conf/httpd.conf. This parameter is critical to IHS performance and must be tuned properly by the IHS administrator. Once IHS reaches the maximum number of Httpd processes allowed by MaxClients, response times for the web site will rise. An administrator can usually determine how many Httpd Daemons are running on a Unix server by issuing the following command:

```
ps –ef | grep httpd | wc –l
```

Determining the proper amount for MaxClients is critical. Experience has shown that a Web Server can undergo bottlenecks on many factors such as memory, CPU, and network. It is best for performance to keep all Httpd Daemons residing in real memory and not raise CPU utilization by having the operating system swap processes to disk. Included within each Httpd process is a TCP Cache that caches static pages. That means that each Httpd Daemon can take up to 10MB worth of real memory. Thus if MaxClients are set to 1024, the web server could consume up to 1GB of memory.

6.6 CPU, Memory and Network Utilization

As mentioned in the preceding section above, monitoring and understanding memory and CPU utilization of the web server and application server are a must. If the web server or application server exceeds the amount of real memory that is available in the hardware then the operating system will attempt to page processes out to disk. This operation can dramatically affect performance of the servers.

6.7 Monitor the Application Server's standard out and standard err logs.

Monitoring the standard out and standard error logs are crucial during a stress test. If some strange behavior is happening during a test then there are usually some clues in the standard out or standard error logs. Depending upon whether an application has a logging framework or not, the application may have its own log or just write to WebSphere's standard out log.

7 Application Profiling

ITA uses JProbe to do application profiling. JProbe captures application run time behavior and help developers to identify performance bottlenecks. JProbe identifies the loitering objects, unused code, and memory usage in an application. The following is a list of the best practices for conducting performance profiling:

7.1 Profiling in Stand-alone Environment

The application should not share a development and a profiling environment. The code should be stable while profiling an application. Also, an application server needs to be restarted often throughout the whole profiling session which might hinder other developers working in the same environment.

7.2 Restart Application Server before each Session

Different profiling sessions have different test goals and criteria. To ensure that a representation of the application's behavior is captured, it is recommended to restart the application server before each session. Restarting the application server before each session guarantees a clean application server environment with resources fully available for a common starting point. A common starting point also serves as a controlled variable that allows developers to monitor the effects of the changes that are made.

7.3 Form Test Scenarios

Profiling tools provide testers with a picture of an application's run-time behavior. They help to identify problem areas. It is the tester's knowledge of the application that locates a potential bottleneck of the application. Thus, it is important for testers to understand the normal memory usage and performance behavior and use this knowledge to form test scenarios including expected end results. Using test scenarios, out-of-ordinary (negative testing) phenomena can be identified at the end of each session and serve as the starting point for further performance tuning.

7.4 Garbage Collect and Establish Baseline

It is important to establish a baseline within each session to see the effects of the test harness. Part of setting the baseline is to execute a garbage collection. By doing so, testers can see which objects are created and destroyed during the profiling session. The left over objects can then be compared to the tester's expectation to determine whether a memory leak has occurred.

7.5 Profiling over Extend Periods of Time

A memory leak problem is not noticeable until the application is run for an extended period of time. It is, therefore, recommended to run test harnesses for at least an hour in order to extract a reasonable representation of memory usage. This can also be achieved by using Load Runner and running the test for an hour.

7.6 Using "–verbosegc"

"-verbosegc" is a command line argument that can be switched on by the testers to log garbage collection in application's "stderr.log file." It is also another way to monitor memory usage during the profiling session.

8 References

1. http://www.mercuryinteractive.com "ActiveTune Production Tuning Service".
2. http://www7b.software.ibm.com "Design Pages for Performance".
3. http://www7b.software.ibm.com "Managing Web Site Performance".
4. http://www.sitraka.com/jprobe